Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 00 (2016) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

25th International Meshing Roundtable

Watertight and 2-manifold surface meshes using Dual Contouring with tetrahedral decomposition of grid cubes

Tanweer Rashid[a], Sharmin Sultana[a], Michel A. Audette[a]*

[a]*Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA 23529, USA*

Abstract

The Dual Contouring algorithm (DC) is a grid-based process used to generate surface meshes from volumetric data. The advantage of DC is that it can reproduce sharp features by inserting vertices anywhere inside the grid cube, as opposed to the Marching Cubes (MC) algorithm that can insert vertices only on the grid edges. However, DC is unable to guarantee 2-manifold and watertight meshes due to the fact that it produces only one vertex for each grid cube. We present a modified Dual Contouring algorithm that is capable of overcoming this limitation. Our method decomposes an ambiguous grid cube into a maximum of twelve tetrahedral cells; we introduce novel polygon generation rules that produce 2-manifold and watertight surface meshes. We have applied our proposed method on realistic data, and a comparison of the results of our proposed method with results from traditional DC shows the effectiveness of our method.

Peer-review under responsibility of the organizing committee of IMR 25.

Keywords: Surface meshing, 2-manifold; watertight; dual contouring; mesh generation; tetrahedral decomposition

1. Introduction

Surface meshing is an invaluable tool and one of the most commonly used methods in scientific research for visualizing volumetric data. A surface mesh of a real-world object can be generated in one of two ways: (1) by using a scanning device such as the NextEngine 3D Laser Scanner or Microsoft's Kinect, or (2) by isosurface extraction from volumetric data such as MRI or CT using contouring algorithms such as *Marching Cubes* (MC) [1], *Dual Contouring* (DC) [2] or *Dynamic Particle Systems* [3]. In both cases, the resulting polyhedral mesh may contain geometric errors such as non-manifold edges and/or vertices, holes and intersecting polygons, especially if the surface

* Corresponding author. Tel.: +1-757-683-6940; fax: +1-757-683-3200.
E-mail address: maudette@odu.edu

Peer-review under responsibility of the organizing committee of IMR 25.

being meshed is complex. The survey of Ju in [4] discusses the wide range of techniques that have been developed for repairing polygonal models.

Non-manifold geometry is problematic for a variety of situations, such as rendering of refractive surfaces, computation of surface normals and curvatures, bounding tetrahedral meshes suitable for finite element analysis and fluid simulations, as well as CAD-based manufacturing and 3D printing. The repairing of geometric errors in meshes is an active research area and there is no one-fits-all algorithm that can fix all the different types of geometric errors. Of course, this is not to say that topologically and geometrically correct surface mesh generation is a poorly researched field. Reference [5] presents an extensive review of the many variants of the MC algorithm that have been developed over the years. *Tight Cocone* [6] is another meshing algorithm that guarantees watertight meshes. *Marching Tetrahedra* [7] is another method similar to MC that can produce topologically correct meshes.

This work focuses primarily on surface meshing with Dual Contouring. DC offers the advantage of producing meshes with sharp features. In MC, the newly created vertices are constrained to the edges of the grid while in DC, the vertices can be anywhere inside the grid cube. However, the traditional DC algorithm produces non-manifold edges and vertices in certain situations. In this work, we present a modified Dual Contouring algorithm that is capable of generating watertight and 2-manifold meshes and thereby avoid non-manifold geometric errors in the first place.

The remainder of this paper is divided into the following sections: Section 2 discusses in general how the traditional DC algorithm works and what the current state of the art is. Section 3 describes our proposed solution in detail. Section 4 describes some of the results of the proposed method and Section 5 discusses the advantage of having geometrically correct surface meshes for tetrahedral mesh generation. Section 6 concludes with a discussion of some of the limitations of the proposed method.

2. Dual Contouring

2.1. An overview of Dual Contouring

Dual Contouring (DC) is a method used for extracting the surface boundary of an implicit volume. The method is dual in the sense that vertices generated by DC are topologically dual to faces in the Marching Cubes (MC) algorithm. In DC, a uniform grid is superimposed on the implicit volume. The grid cubes are represented as nodes in an octree data structure. For each grid cube intersecting the volume, the eight corners of the cube are assigned inside/outside labels, and a quadratic error function (QEF) is defined as:

$$E[x] = \sum (N_i \cdot (x - p_i))^2 \tag{1}$$

where x is the computed dual vertex or *minimizer*, and p_i and N_i represent the intersections and unit normal, respectively, of the volume boundary with the edges of the cube.

Fig. 1 illustrates the basic concept of QEFs in 2D. The bounding surface of the volume shown in light blue color intersects the lower left corner of a square. The lower left corner of the square is marked with a “+” sign indicating that it lies inside the volume while the remaining corners of the square are marked with a “-” sign indicating that they lie outside the volume. Furthermore, the surface intersects the left and bottom edges of the square at points p_0 and p_1, respectively.

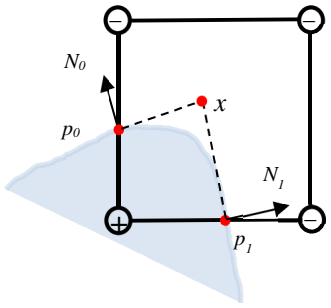


Fig. 1. Formulation of Quadratic Error Functions. The blue region represents the surface/volume.

If a tangent is drawn from points p_0 and p_1 and extended inside the square, they would intersect each other somewhere inside the square at x. This point would be a vertex of the isosurface. Typically, one minimizer is computed for each grid cube (in the 3D case) containing a sign change. The minimizer can be anywhere inside the square or cube, rather than being restricted to the edges of the square or cube as in MC. This feature allows DC to produce meshes with sharp features. The objective function $E[x]$ can be expressed as the inner product $(Ax-b)^T(Ax-b)$ where A is a matrix whose rows are the normals N_i and b is a vector whose entries are $(N_i \cdot p)$. The function $E[x]$ can then be expanded as

$$E[x] = x^T A^T Ax - 2x^T A^T b + b^T b \tag{2}$$

where $A^T A$ is a symmetric 3x3 matrix, $A^T b$ is a column vector of length three and $b^T b$ is a scalar. This representation of a QEF can be solved using the QR decomposition [8], and it should be noted that Singular Value Decomposition (SVD) [9, 10] can also be employed for solving this system.

In traditional DC, a recursive method using the three recursive functions *cellProc()*, *faceProc()* and *edgeProc()* is used to traverse through the octree during the polygon generation phase. For each minimal edge exhibiting a sign change, a quad or two triangles are generated by connecting the minimizers of the four cubes containing the minimal edge.

2.2. Background and literature review

One of the main disadvantages of DC is that it does not guarantee 2-manifold and intersection-free surfaces. A polygonal mesh is considered as being 2-manifold if each edge of the mesh is shared by only two faces, and if the neighborhood of each vertex of the mesh is the topological equivalent of a disk. Ju and Udeshi address the issue of intersecting triangles in [11] by proposing a hybrid method where dual vertices (inside grid cubes) as well as face vertices and edge vertices (inserted on the cube's face and edges, respectively) are used to create polygons according to new polygon generation rules. Zhang et al. in [12] present a topology-preserving algorithm for surface simplification using vertex clustering and an enhanced cell representation, but this method is unable to avoid non-manifold edges and vertices. Varadhan et al. [13] suggest an approach that combines edge intersection testing, adaptive subdivision, and dual contouring to reconstruct thin features. Schaefer et al. use a vertex clustering method in [14], where they present an additional topology criterion that must be satisfied to ensure manifoldness.

Zhang and Qian in [15] take a different approach by first generating a base mesh using standard DC, and then analyzing and categorizing the octree leaf cells into 31 topology groups. For ambiguous cubes, multiple minimizers, as many as three in some instances, are inserted whereby a new topologically correct mesh is created by reconnecting the vertices of the mesh with the newly inserted minimizers. In [16], Zhang and Qian decompose ambiguous cubes into twelve tetrahedral cells, each having one minimizer, and construct a series of polygons and polyhedrons to create tetrahedral meshes. This method can avoid topological ambiguities in tetrahedral meshes but does not produce surface meshes.

Our proposed method uses an approach similar to that in [16] by decomposing an ambiguous cube into several tetrahedral cells. In this work, we introduce novel polygon generation rules that result in 2-manifold and watertight triangular surface meshes.

3. Watertight and 2-manifold Dual Contouring

Our proposed method begins the same way as in traditional Dual Contouring (DC) by superimposing a uniform virtual grid onto the implicit volume. Depending on the isovalue chosen, the corners of each cube can have 2^8 or 256 possible configurations. By taking rotation and symmetry into account, these configurations can be reduced into 14 fundamental cases, as shown in Fig. 2. Cases 0, 1, 2, 5, 8, 9 and 11 are simple unambiguous cases, meaning there is only one possible surface intersecting the grid cube (no surface for Case 0). Cases 3, 4, 6, 7, 10, 12 and 13 are ambiguous, meaning that there is more than one possible surface that intersects the cube. It is the presence of these ambiguous cubes, as well as the fact that traditional Dual Contouring produces only one minimizer for each cube, that causes non-manifold surfaces to arise. Additionally, in our experiments we have observed that the complement of

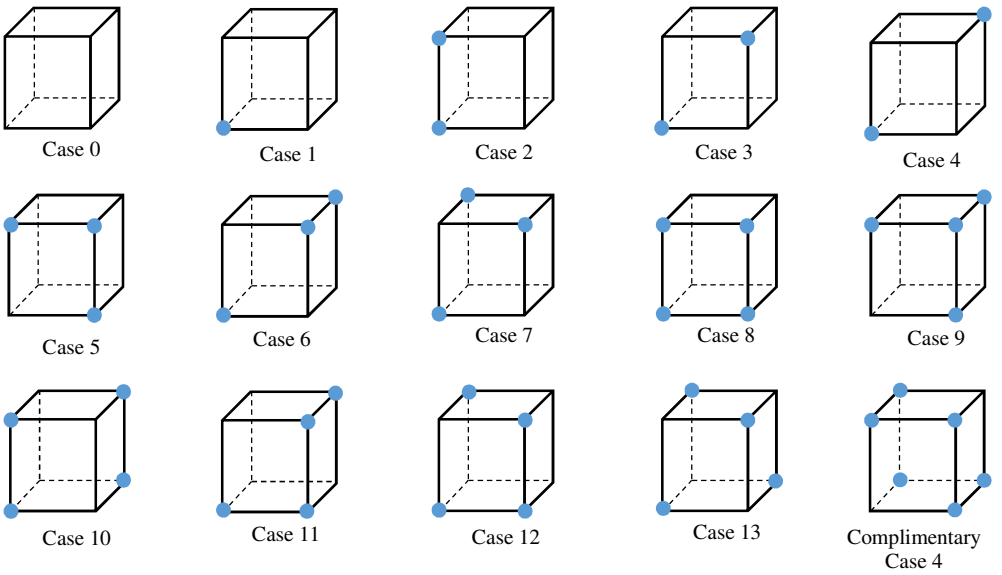
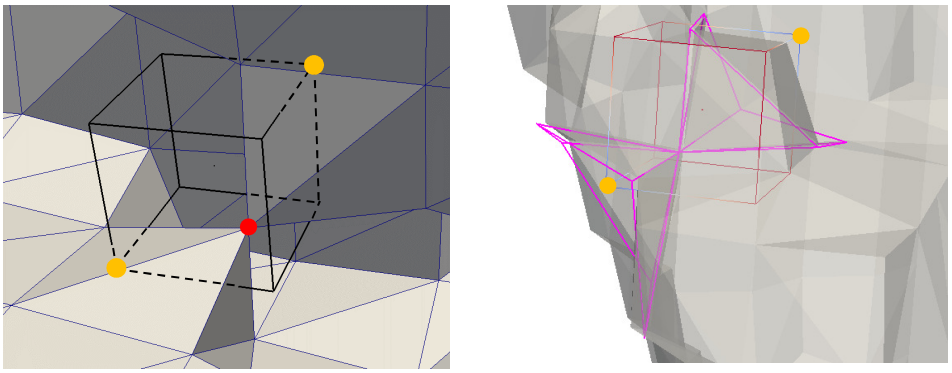


Fig. 2. The 14 fundamental configurations for a grid cube, as well as complimentary Case 4.

Case 4 (that is, a situation where the two diagonally opposite corners of the cube are in background and the rest are in the foreground) is also responsible for the generation of non-manifold vertices, as shown in Fig. 3. These particular non-manifold vertices occur inside the surface mesh. In [17], Sohn shows that a cubic cell can be decomposed into a set of tetrahedral cells while preserving the topology of the isosurface inside the cube. However, the number of tetrahedral cells created, as well as their shapes and sizes, is dependent on the number of face and body saddle points. The tetrahedral decomposition method in [16] is much more advantageous because any ambiguous cube can be decomposed into 12 tetrahedral cells of similar shape and size.

3.1. Tetrahedral decomposition of ambiguous cubes

The volume is first subdivided into a uniform grid of an appropriate size. An octree whose leaves represent the grid cubes is then used for fast parsing of the grid cubes and polygon generation. Each corner of a grid cube is assigned an



(a) (b)

Fig. 3. Non-manifold vertex due to complimentary Case 4. (a) This is an inside view of the surface mesh with the non-manifold vertex highlighted by the red dot. (b) This is a view from outside the surface mesh (rendered transparent). The highlighted polygons share the non-manifold vertex. The two yellow dots represent the two diagonally opposite corners of the cube and they lie outside the implicit volume.

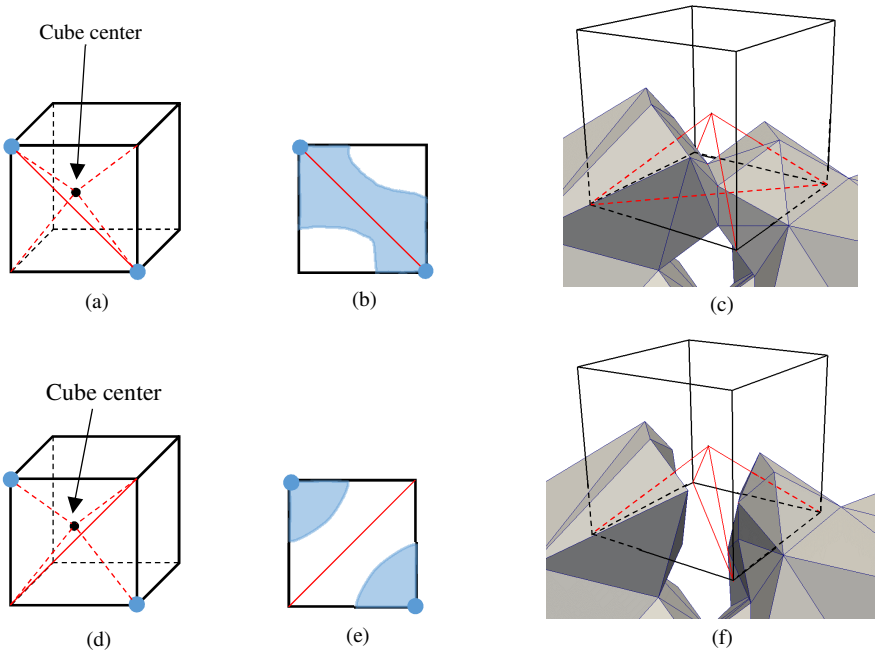


Fig. 4. (a, d) Two ways in which a diagonal can be created on the front-most face of an ambiguous cube to generate two tetrahedra. (b) The corners of the cube are contiguously inside the volume, (e) the corners of the cube are inside the volume, but not contiguously (c, f). Two differing topologies can occur due to different choice of diagonals.

inside or outside label based on its location in the implicit volume. For each unambiguous cube, one minimizer is computed.

In the case of ambiguous cubes, we follow an approach similar to that of Zhang and Qian in [16] and the ambiguous cube is subdivided into a maximum of twelve tetrahedral cells. The center of the cube is a common point for all the newly created tetrahedral cells, as shown in Fig. 4(a) and (d). A tetrahedral cell is made up of the center along with three corners of a cube face. Each face of the ambiguous cube forms the base of two tetrahedral cells by joining the diagonally opposite corners of the face. Fig. 4(b) and (e) illustrates two ways for creating the diagonal, which is further explained below:

- If the two diagonally opposite corners are contiguous with each other through the interior of the face, then create a diagonal between the two corners (Fig. 4b).
- If the two diagonally opposite corners are inside the volume, but not contiguous with each other through the interior of the face, then create a diagonal using the other two corners (Fig. 4e).
- For all other cases, any appropriate diagonal can be used.

The choice of creating the diagonal is important because the resulting polygonization can lead to topological changes. Fig. 4(c) depicts a situation where the two corners of the bottom face of the ambiguous cube are contiguous with each other inside the face, and the diagonal is created as shown in Fig. 4(b). Fig. 4(f) shows a situation where the two corners of the bottom face of the ambiguous cube are inside the volume, but not contiguous through the interior, and the diagonal for the bottom face is created as in Fig. 4(e).

It is useful to define a few terms at this point. An *interior edge* is an edge of a tetrahedron that is made up of a corner of the parent cube and the center of the cube. A *sign change edge* is an edge of a tetrahedron or a grid cube whose end points have inside and outside labels. A *valid* tetrahedron is one in which at least one edge is a sign change edge. Our proposed strategy makes use of only valid tetrahedra for polygon generation.

During polygon generation, the centroid of each tetrahedron is used as a minimizer. The method of Zhang and Qian in [6] presents two general rules that generate tetrahedra/polygons using the minimizers of unambiguous grid cubes

as well as the minimizers of the tetrahedral cells from ambiguous grid cubes. The effect of incorporating these rules into the traditional DC algorithm results in a mesh comprising of both surface and tetrahedral meshes.

In our strategy, we present novel rules that result in purely surface meshes, rather than tetrahedral meshes. Our method is effective in generating 2-manifold meshes and can be easily incorporated into the DC algorithm. We present the details of our rules in the next section.

3.2. Polygon generation

During the polygon generation phase, we follow the traditional DC approach by analyzing minimal edges as well as other sign change edges. Each minimal edge is an edge that is characterized by a sign change and that is also shared between four grid cubes. If all four grid cubes sharing that edge are unambiguous, then we create two triangles using the four minimizers of the four grid cubes. On the other hand, if any one of the four incident grid cubes is ambiguous, we apply the following rules: (1) Minimal Edge Rule, (2) Face Diagonal Rule, and (3) Interior Edge Rule as described below:

Minimal Edge Rule: Create an n-sided polygon, or n-gon, using the minimizers of all the unambiguous cubes and tetrahedral cells that contain the minimal edge.

This rule follows the same concept as in traditional DC: if the minimal edge is a sign-change edge, then there must be surface intersecting the minimal edge. We generate the n-sided polygon by linking together the minimizers of unambiguous grid cubes and tetrahedral cells that share the minimal edge, and then triangulating the n-gon. Each ambiguous grid cube will have exactly two valid tetrahedral cells sharing the minimal edge. It should be noted that the resulting n-gon does not necessarily have to be convex. It is also worth mentioning that extra care should be taken when using third-party polygon generation algorithms, such as Delaunay tessellation, which have a tendency to generate convex polygons. Fig. 5 illustrates the Minimal Edge Rule. In this figure, the black square represents one end of the minimal edge that is shared by the four grid cubes. There are two ambiguous grid cubes (colored green) and two unambiguous grid cubes (colored grey). The small blue and red squares represent the vertices of the tetrahedral cells. The blue and red lines represent the four tetrahedral cells of the two ambiguous grid cubes. All four tetrahedral cells share the minimal edge. In Fig. 5, a 6-sided polygon is created by first linking together the minimizers of the two unambiguous grid cubes (green circles), as well as the minimizers of the four tetrahedral cells (red and blue circles) sharing the minimal edge, and then triangulating the polygon.

Face Diagonal Rule: For any ambiguous cube sharing a face with another cube, if the face diagonal of the shared face is a sign change edge, then create a polygon using all the minimizers surrounding the face diagonal.

Ambiguous grid cubes can share a face with another ambiguous or unambiguous grid cube. In the case of two ambiguous cubes sharing a face, if the face diagonal is a sign change edge, then there are four valid tetrahedral cells that share the face diagonal. We use the four minimizers to generate a quad, which equates with two triangles.

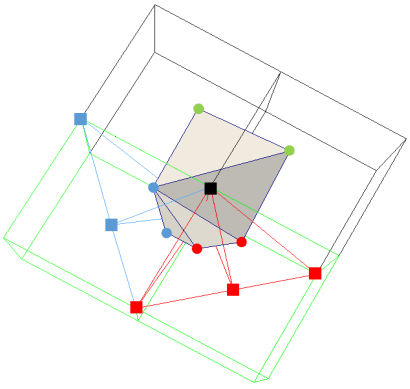


Fig. 5. An illustration of the Minimal Edge Rule. The two grey cubes are unambiguous cubes and the two green cubes are ambiguous cubes. The black square represents one end of the minimal edge. The blue and red squares represent the vertices of the tetrahedral cells. The two blue and two red circles represent the minimizers of the tetrahedra incident on the minimal edge.

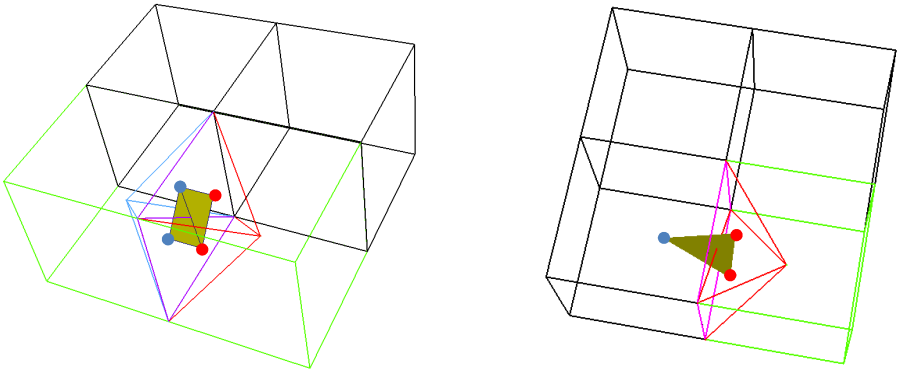


Fig. 6. An illustration of the Face Diagonal Rule. The purple square is the shared face between two cubes. Ambiguous cubes are colored green and unambiguous cubes are colored black. (a) The case of two ambiguous cubes sharing a face. (b) The case of an ambiguous cube and an unambiguous cube sharing a face.

In the case of an ambiguous cube sharing a face with an unambiguous cube, there are two valid tetrahedral cells whose bases comprise the shared face. We use the minimizers of these two tetrahedral cells as well as the minimizer of the unambiguous cube and generate a triangle. Fig. 6 illustrates the application of this rule. In Fig. 6(a), the two green cubes are the ambiguous cubes while the red and blue lines represent the four tetrahedral cells sharing the face diagonal. The face diagonal is a sign change edge. The black cubes are unambiguous. The purple square represents the shared face between the two ambiguous cubes. The red and blue round points represent the four minimizers used to generate the two yellow triangles. In Fig. 6(b), the green cube is the sole ambiguous cube and the red lines represent the two tetrahedral cells making up the shared face (purple square) with a neighboring unambiguous cube. The two red round points are the minimizers of the two tetrahedral cells and the blue round point is the minimizer of the unambiguous cube in question. The three minimizers are used to generate a triangle (yellow).

Interior Edge Rule: For a sign-change interior edge of a tetrahedral cell which has one end point that is also shared with the minimal edge, create a polygon using the minimizers of all the tetrahedral cells of the parent cube that share the sign-change interior edge.

An interior edge can be shared by multiple tetrahedral cells. If the interior edge is a sign-change edge, then it follows that there is a surface intersecting the interior edge, and this surface can be constructed using the minimizers of the surrounding tetrahedral cells. Fig. 7 depicts the Interior Edge Rule. In this figure, there is one ambiguous grid

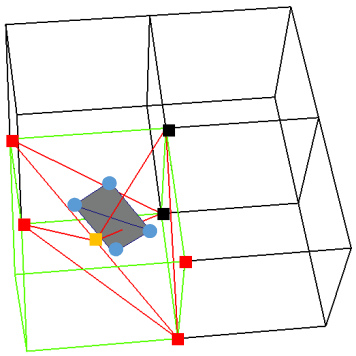


Fig. 7. An illustration of the Interior Edge Rule. The black squares represent the minimal edge. The red squares depict the vertices of the four tetrahedral cells incident on the minimal edge: the vertices of the minimal edge are also vertices of two tetrahedral cells. The bottom black square and orange square make up the sign-change interior edge. The red lines represent the four tetrahedral cells that share the sign-change edge. The blue round points are the minimizers of the tetrahedral cells that are used to create a polygon.

cube (green) and three unambiguous grid cubes (grey). The center of the ambiguous cube (orange square) is a shared vertex for all the tetrahedral cells. The small red squares are the vertices of the tetrahedral cells. The two black squares make up the minimal edge. The interior edge with a sign-change edge in this figure is made of the bottom black square and the center of the cube (orange square). The red lines represents four tetrahedral cells that share the sign change interior edge. The blue round points are the minimizers of the tetrahedral cells. A polygon is created using these four minimizers.

3.3. Detection of non-manifold edges and vertices and boundary edges

The proposed method is a modified Dual Contouring algorithm that can produce 2-manifold and watertight surface meshes. In the course of this work, we did not rely only on visual inspection to detect the presence or absence of non-manifold edges and vertices and boundary edges. We used MeshLab [18] to detect the presence or absence of non-manifold edges and vertices and boundary edges. MeshLab is an open source mesh processing tool that makes extensive use of the VCG (Visualization and Computer Graphics) Library (http://vcg.isti.cnr.it/vcglib/).

4. Results and discussion

We have presented a modified version of the Dual Contouring algorithm that is capable of overcoming some of the limitations of traditional DC. Our proposed method uses tetrahedral decomposition of ambiguous cubes to generate 2-manifold and watertight surface meshes.

Fig. 8(a) shows a non-manifold vertex that was created due to a cube exhibiting the complimentary Case 4 configuration using traditional DC. Fig. 8(b) shows the 2-manifold tube-like mesh replacing the non-manifold vertex using our proposed method. Fig. 8(c) shows another example of a non-manifold vertex created using traditional DC, and Fig. 8(d) shows that it has been replaced by a mesh that is separate and not tubular using the proposed method.

We have also applied our proposed algorithm on a digital atlas [19] of deep brain structures, specifically the basal ganglia and thalamus, created using serial histological data. This atlas is in MINC 2.0 (Medical Imaging NetCDF) format and contains a total of 123 labeled structures. The atlas consists of 334x334x334 voxels with a stepsize of 0.3mm.

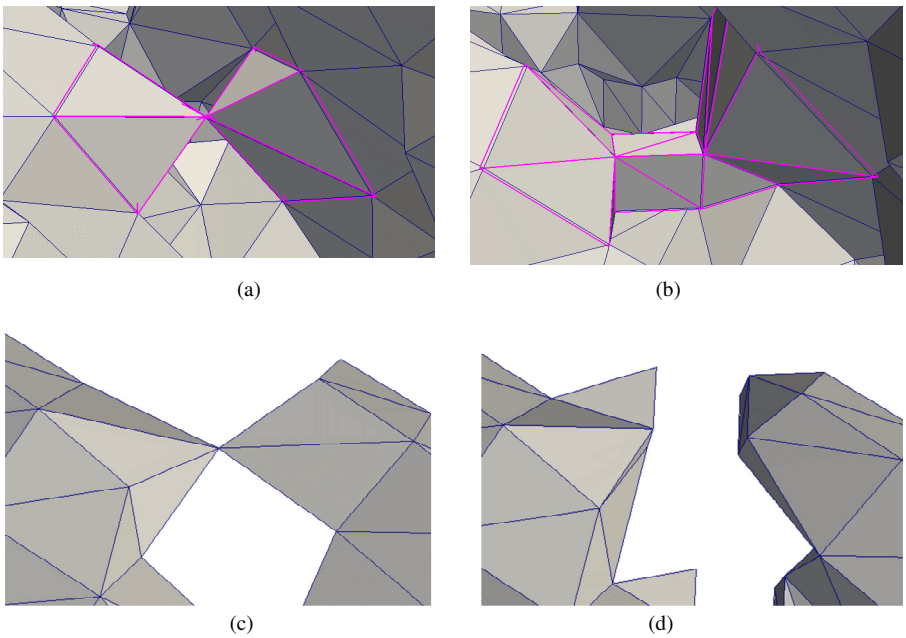


Fig. 8. Two examples of a non-manifold vertex being replaced by a 2-manifold mesh. (a, c) shows two different non-manifold vertices generated using traditional DC, (b, d) shows their corresponding 2-manifold solutions generated using the proposed method.

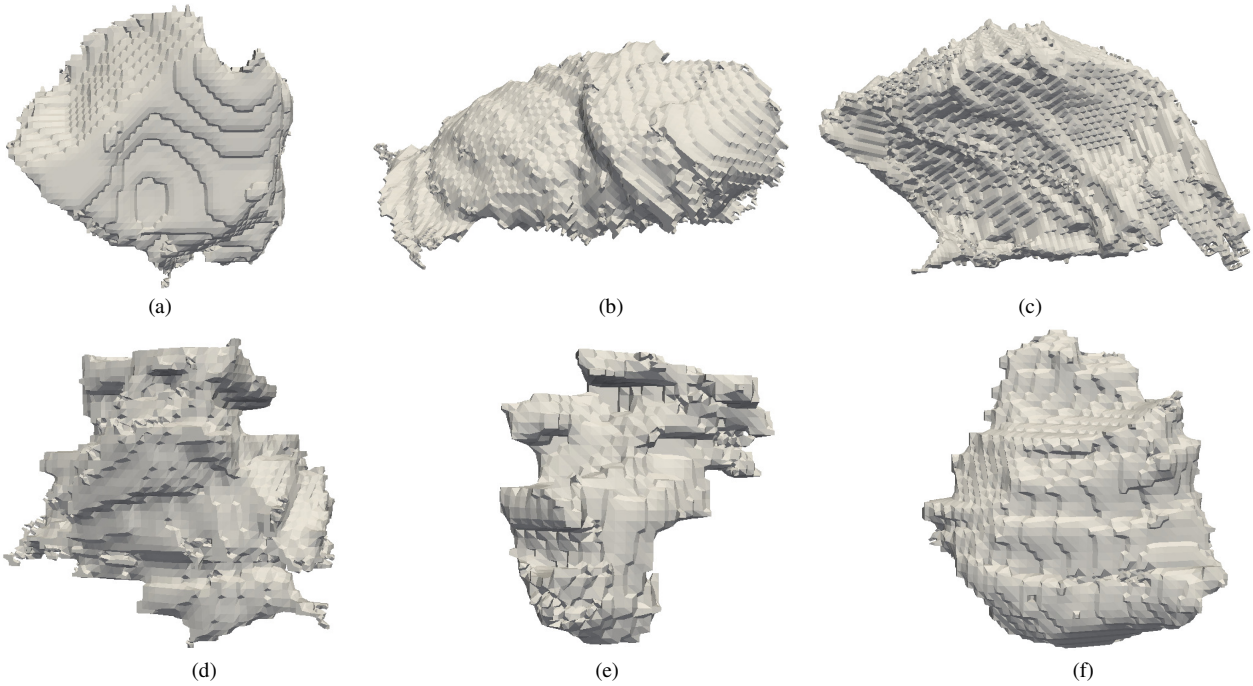


Fig. 9. Surfaces generated from a digital deep brain atlas using the proposed method without morphological preprocessing. (a) Globus Pallidus (b) Globus Pallidus Internal (c) Globus Pallidus External (d) Nucleus lateropolaris thalami (e) Nucleus fasciculosus thalami (f) Subthalamic Nucleus.

We have applied both traditional DC and our proposed method to generate surface meshes of deep brain structures in two experiments. In the first experiment, the atlas data was not preprocessed in any way. Fig. 9 shows the resulting surface meshes generated by the proposed method. In all cases, the resulting meshes are 2-manifold and watertight. Table 1 lists the names of the atlas labels and their corresponding number of vertices and triangles and number of non-manifold edges and vertices for both traditional DC and the proposed method. The number of vertices and triangles for meshes generated by the proposed method is significantly higher than the corresponding meshes generated using traditional DC. This is because the proposed method inserts multiple minimizers (as many as twelve in some cases) in ambiguous cubes using tetrahedral decomposition, and results in an increase in the number of triangles and vertices.

We performed a second experiment on the same deep brain structures using both traditional DC and the proposed method. For this experiment, some preprocessing was performed on the atlas data: Each deep brain structure was separated and binarized into a single volume. A crude smoothing was performed using the binary morphological operations opening and closing. The structuring element was a cube of size 3x3x3. The sequence of morphological

Table 1. A comparison between traditional DC and the proposed method using unpreprocessed data

			Traditonal DC		Proposed method	
	Atlas label	Structure name	Number of vertices/tris	Number of non-manifold edges/vertices	Number of vertices/tris	Number of non-manifold edges/vertices
Fig. 9(a)	05	Globus Pallidus	26068/53460	656/70	35750/72670	0/0
Fig. 9(b)	11	Globus Pallidus Internal	23396/48088	642/67	32369/66018	0/0
Fig. 9(c)	12	Globus Pallidus Externa	26408/53644	453/39	32707/66226	0/0
Fig. 9(d)	26	Nucleus lateropolaris thalami	13401/27424	333/31	18123/36806	0/0
Fig. 9(e)	27	Nucleus fasciculosus thalami	4971/10096	82/9	6231/12490	0/0
Fig. 9(f)	39	Subthalamic Nucleus	9939/20152	148/8	12006/24276	0/0

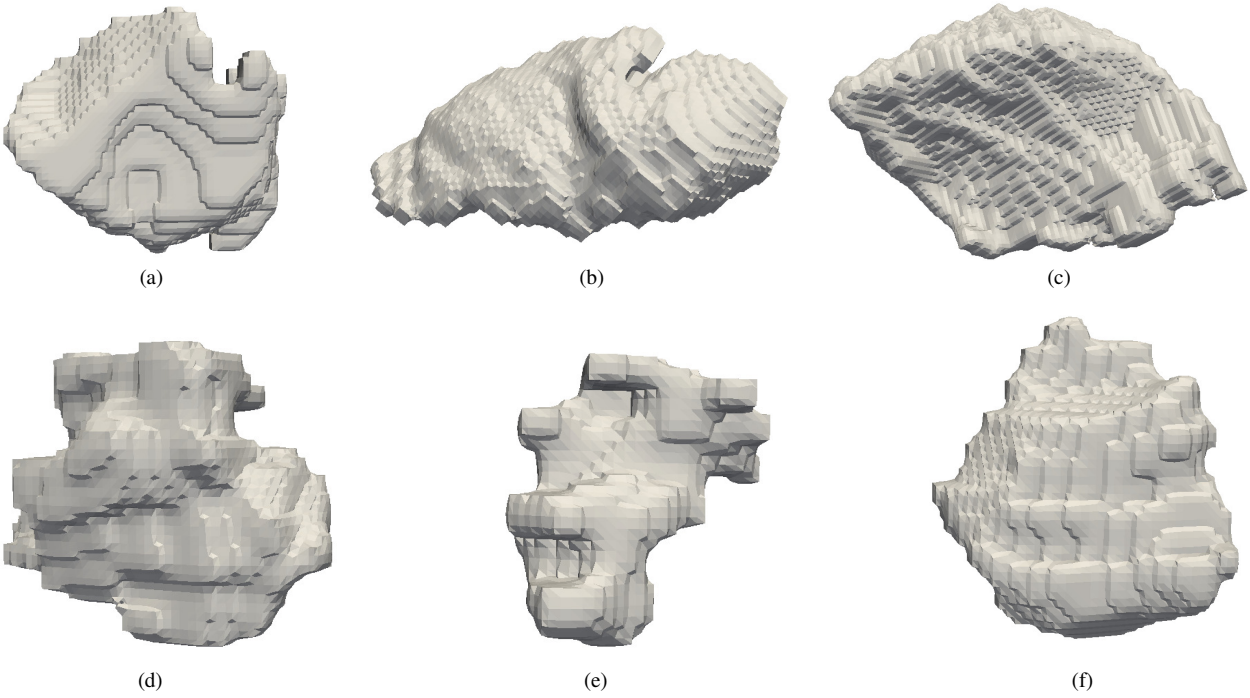


Fig. 10. Surfaces generated from a digital deep brain atlas using the proposed method with morphological preprocessing. (a) Globus Pallidus (b) Globus Pallidus Internal (c) Globus Pallidus External (d) Nucleus lateropolaris thalami (e) Nucleus fasciculosus thalami (f) Subthalamic Nucleus. The different structures were separated and binarized into individual volumes, and then preprocessed using morphological operations.

operations were closing, followed by opening. This preprocessing was done to considerably simplify the topology of the volume. Fig. 10 shows the resulting meshes generated by the proposed method. Table 2 lists the names and labels of the structures, as well as the number of vertices and triangles, and the number of non-manifold edges and vertices for the meshes of both traditional DC and the proposed method.

The purpose of this second experiment is to show that even after preprocessing, traditional DC is unable to produce 2-manifold meshes in almost all cases, with Fig. 10(e) being the exception. Preprocessing using morphological operations greatly simplifies the topology of the surface. That is why there are fewer non-manifold edges and vertices in the meshes generated using traditional DC. The simplified volume therefore contains fewer ambiguous cubes. So the number of vertices and triangles in the meshes generated by the proposed method are not significantly higher than the number of vertices and triangles of the meshes generated by traditional DC. In the case of Fig. 10(e), the preprocessing operation simplified the volume to such an extent that there were no ambiguous cubes present, and therefore, the proposed method behaved exactly like traditional DC. For the other structures, the proposed method

Table 2. A comparison between traditional DC and the proposed method using preprocessed data

			Traditonal DC		Proposed method	
	Atlas label	Structure name	Number of vertices/tris	Number of non-manifold edges/vertices	Number of vertices/tris	Number of non-manifold edges/vertices
Fig. 10(a)	05	Globus Pallidus	16858/33712	2/0	16890/33776	0/0
Fig. 10(b)	11	Globus Pallidus Internal	15736/31480	7/0	15847/31698	0/0
Fig. 10(c)	12	Globus Pallidus Externa	20411/40828	7/0	20506/41016	0/0
Fig. 10(d)	26	Nucleus lateropolaris thalami	9608/19212	3/0	9657/19314	0/0
Fig. 10(e)	27	Nucleus fasciculosus thalami	3550/7092	0/0	3550/7092	0/0
Fig. 10(f)	39	Subthalamic Nucleus	8165/16324	1/0	8181/16358	0/0

Table 3. Mesh quality for meshes generated by the proposed method using unpreprocessed data and preprocessed data.

		Meshes from unpreprocessed data		Meshes from preprocessed data	
Atlas label	Structure name	Average Radius ratio (Best/Worst)	No of tris (No of tris with radius ratio <= 0.2)	Average Radius ratio (Best/Worst)	No of tris (No of tris with radius ratio <= 0.2)
05	Globus Pallidus	0.744 (1.0/0.00060288)	72670 (503)	0.782 (1.0/0.0629017)	33776 (102)
11	Globus Pallidus Internal	0.741 (1.0/0.00013993)	66018 (482)	0.781 (1.0/0.0111115)	31698 (82)
12	Globus Pallidus Externa	0.747 (1.0/0.00012822)	66226 (487)	0.775 (1.0/0.0461131)	41016 (144)
26	Nucleus lateropolaris thalami	0.745 (1.0/0.00527279)	36806 (271)	0.781 (1.0/0.0122761)	19314 (78)
27	Nucleus fasciculosus thalami	0.756 (1.0/0.00026428)	12490 (81)	0.784 (1.0/0.0002641)	7092 (29)
39	Subthalamic Nucleus	0.747 (1.0/0.00241204)	24276 (155)	0.778 (1.0/0.0246822)	16358 (60)

generated meshes that are completely 2-manifold and watertight.

Fig. 11 shows all the anatomical structures of Fig. 9 superimposed over an XY slice of the digital deep brain atlas. The atlas itself is rendered in gray scale.

A commonly used metric to describe the quality of triangles in surface meshes is the radius ratio $2r_{in}/r_{circ}$ [20] where r_{in} is the radius of the circle inscribed in the triangle, and r_{circ} is the radius of the circumscribing circle. A value close to 1 indicates a very good quality triangle (similar to an equilateral triangle) and a value near zero indicates a poor quality triangle (a triangle which is collapsing to an edge). The average values for the structures in Fig. 9 and Fig. 10 are reported in Table 3. Also included in Table 3 are the number of triangles whose ratio values fall below a threshold (0.2 in this case), as well as the best and worst values. As can be seen, there are a number of triangles whose radius ratios are less than ideal. Most of these poor quality triangles are created by the Minimal Edge Rule. As mentioned before, the *n*-gon created in this rule does not necessarily have to be convex. In our implementation, we used an ad-hoc method to triangulate the *n*-gon but this triangulation method is not configured towards producing high quality triangles as Delaunay-based methods do. However, Delaunay-based tessellation methods tend produce convex triangulations and enforcing the *n*-gon to be always convex can introduce non-manifold edges in the resulting surface mesh.

5. Tetrahedral mesh generation

One type of mesh that is commonly used in engineering and biomedical research is the tetrahedral mesh. Tetrahedral mesh generation can be classified [21] into the following four categories: (1) Octree-based, (2) Delaunay, (3) Advancing Front and (4) Optimization-based. Among these categories, Delaunay based techniques are the most frequently used. In many applications, an initial surface mesh, known as a piecewise linear complex (PLC) that

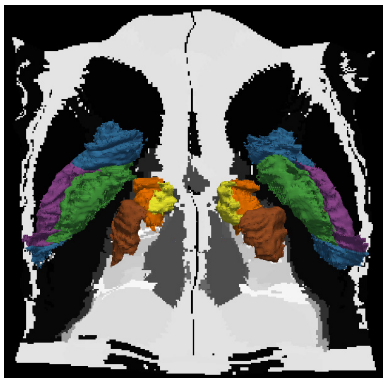


Fig. 11. The anatomical structures of Figure 10 superimposed over a slice along the XY plane (Z = 163) of the digital atlas. The atlas is depicted in gray scale, and the mesh coloring is as follows: Globus Pallidus (blue), Globus Pallidus Internal (green), Globus Pallidus External (purple), Nucleus lateropolaris thalami (orange), Nucleus fasciculosus thalami (yellow) Subthalamic Nucleus (brown).

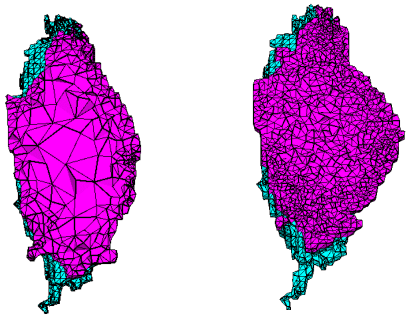


Fig. 12. Tetrahedral meshes created using the surface mesh of the Nucleus lateropolaris thalami (atlas label 26) as the input PLC. (Left) coarse tetrahedralization (Right) Finer tetrahedralization.

coincides with the boundary of the problem domain is used as an initial starting point for the tetrahedralization process. In such cases, the user has to ensure (either manually or by using mesh editing software) that the input surface mesh does not contain geometric errors such as holes, slivers, intersecting triangles and non-manifold elements.

Software such as TetGen [22] and the opensource library CGAL (Computational Geometry Algorithms Library) [23] are able to generate tetrahedral meshes from an input PLC using Delaunay-based methods. Fig. 12 shows two cross-sections of tetrahedral meshes generated using TetGen, using the surface mesh of the Nucleus lateropolaris thalami (Fig. 9(d)) as input. The purple colored triangles represent the tetrahedral elements and the blue colored triangles represent the input surface mesh. Fig. 12 (left) shows a coarse tetrahedralization (when the edge-radius ratio is set to 1.5) and Fig. 12 (right) shows a finer tetrahedralization (when the edge-radius ratio is set to 1.0). Although TetGen and CGAL both provide facilities for mesh refinement and optimal tetrahedralization, none of those facilities were utilized during the generation of the meshes in Fig. 12. The main purpose of this section is to emphasize the fact that the surface meshes generated using our proposed methodology are error free (2-manifold, watertight and intersection free) and can be readily used in the generation of tetrahedral meshes.

6. Limitations and conclusions

The traditional Dual Contouring (DC) algorithm can be made to use adaptive as well as non-adaptive octrees. In the adaptive approach, the grid cubes can be of different sizes (different levels of the octree) whereas in the non-adaptive approach, all the grid cubes are of the same size (lowest level of the octree). One of the limitations of our proposed solution is that it is only applicable in non-adaptive octrees where a minimal edge will always have four grid cubes sharing the edge.

Another limitation of our proposed solution is that it assumes that minimizers are always inside their respective grid cubes. This is not an issue in the case of the tetrahedral cells because we use their centroids as minimizers. In the case of grid cubes, the minimizers are computed from Quadratic Error Functions, and the method used to solve these QEFs may be a factor. In certain cases, the QR decomposition or SVD may result in the minimizer being placed outside its grid cube. This allows the resulting mesh to have a smoother appearance, but also results in the mesh containing intersecting triangles and/or cracks. Schmitz et al. uses an iterative particle-based method to compute minimizers in [24] which results in good approximations of the isosurface, but does not guarantee a solution. In our solution, we restrict minimizers to remain within their respective cubes, and this gives the resulting meshes a more blocky appearance.

We have presented a modified Dual Contouring algorithm that is capable of generating 2-manifold surface meshes. Since non-manifoldness occurs due to the presence of ambiguous grid cubes, we proposed a method to subdivide an ambiguous cube into tetrahedral cells. The centroid of these tetrahedral cells are used as minimizers, and allows an ambiguous cube to have multiple minimizers. We have also presented novel polygon generation rules that result in 2-manifold surfaces.

We have presented two sets of results using unpreprocessed data and preprocessed data. For both sets of data, we applied traditional DC and the proposed method to generate surface meshes. We used the facilities in MeshLab, instead

of relying on visual inspection, to confirm the presence or absence of boundary edges and non-manifold edges and vertices. In both cases, our proposed method produced meshes that were 2-manifold and watertight, while traditional DC produced meshes with non-manifold edges and vertices. Our proposed strategy is simple and effective, and can be easily integrated into the traditional Dual Contouring algorithm. The resulting surface meshes are error free, and can easily be used for the generation of tetrahedral meshes.

Acknowledgements

We would like to thank Dr. Powei Feng for providing us with an implementation of a multi-material Dual Contouring algorithm as well as Dr. Tao Ju of Washington University in St. Louis, Missouri, USA for his generous contributions and help. We would also like to thank Dr. Mallar Chakravarty for providing us the digital deep brain atlas for our use. Lastly we would like to thank Dr. Yongjie Zhang of Carnegie Mellon University and Dr. Andrey Chernikov of Old Dominion University for their help.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in ACM Siggraph Computer Graphics, 1987, pp. 163-169.
[2] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in ACM Transactions on Graphics (TOG), 2002, pp. 339-346.
[3] A. P. Witkin and P. S. Heckbert, "Using particles to sample and control implicit surfaces," in Proceedings of the 21st annual conference on Computer graphics and interactive techniques, 1994, pp. 269-277.
[4] T. Ju, "Fixing geometric errors on polygonal models: a survey," Journal of Computer Science and Technology, vol. 24, pp. 19-29, 2009.
[5] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," Computers & Graphics, vol. 30, pp. 854-879, 2006.
[6] T. K. Dey and S. Goswami, "Tight cocone: a water-tight surface reconstructor," in Proceedings of the eighth ACM symposium on Solid modeling and applications, 2003, pp. 127-134.
[7] G. M. Treece, R. W. Prager, and A. H. Gee, "Regularised marching tetrahedra: improved iso-surface extraction," Computers & Graphics, vol. 23, pp. 583-598, 1999.
[8] G. H. Golub and C. F. Van Loan, "Matrix Computations. Johns Hopkins series in the mathematical sciences," Johns Hopkins University Press, Baltimore, MD, 1989.
[9] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp. 57-66.
[10] P. Lindstrom, "Out-of-core simplification of large polygonal models," in Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 259-262.
[11] T. Ju and T. Udeshi, "Intersection-free contouring on an octree grid," in Pacific graphics, 2006.
[12] N. Zhang, W. Hong, and A. Kaufman, "Dual contouring with topology-preserving simplification using enhanced cell representation," in Visualization, 2004. IEEE, 2004, pp. 505-512.
[13] G. Varadhan, S. Krishnan, Y. J. Kim, and D. Manocha, "Feature-sensitive subdivision and isosurface reconstruction," in Visualization, 2003. VIS 2003. IEEE, 2003, pp. 99-106.
[14] S. Schaefer, T. Ju, and J. Warren, "Manifold dual contouring," Visualization and Computer Graphics, IEEE Transactions on, vol. 13, pp. 610-619, 2007.
[15] Y. Zhang and J. Qian, "Dual contouring for domains with topology ambiguity," Computer Methods in Applied Mechanics and Engineering, vol. 217, pp. 34-45, 2012.
[16] Y. Zhang and J. Qian, "Resolving topology ambiguity for multiple-material domains," Computer Methods in Applied Mechanics and Engineering, vol. 247, pp. 166-178, 2012.
[17] B.-S. Sohn, "Topology preserving tetrahedral decomposition of trilinear cell," in Computational Science–ICCS 2007, ed: Springer, 2007, pp. 350-357.
[18] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool," in Eurographics Italian Chapter Conference, 2008, pp. 129-136.
[19] M. M. Chakravarty, G. Bertrand, C. P. Hodge, A. F. Sadikot, and D. L. Collins, "The creation of a brain atlas for image guided neurosurgery using serial histological data," Neuroimage, vol. 30, pp. 359-376, 2006.
[20] J. Shewchuk, "What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)," 2002.
[21] S. J. Owen, "A Survey of Unstructured Mesh Generation Technology," in IMR, 1998, pp. 239-267.
[22] H. Si, "TetGen, a Delaunay-based quality tetrahedral mesh generator," ACM Transactions on Mathematical Software (TOMS), vol. 41, p. 11, 2015.
[23] T. C. Project". (2016). CGAL User and Reference Manual (4.8 ed.). Available: http://doc.cgal.org/4.8/Manual/packages.html
[24] L. Schmitz, C. Dietrich, and J. L. Comba, "Efficient and high quality contouring of isosurfaces on uniform grids," in Computer Graphics and Image Processing (SIBGRAPI), 2009 XXII Brazilian Symposium on, 2009, pp. 64-71.